We would like to thank the reviewers for their time spent crafting

constructive feedback.

We would like to clarify for Reviewer C that, as noted in Section

3.3.6, we check whether \*any\* of the fields (not just "To:" fields) which accept user input we

injected (any data, not just malicious payloads) appeared as part of

\*either the headers or the body\* of the resulting e-mail. If these

user-input-based fields were found on the resulting e-mail, we then

injected these fields with the payloads to check for the presence of

e-mail header injection. The forms that we fuzzed were also not

limited to only forms that allowed users to specify "To:" fields, but

also regular forms.

Reviewer C expressed concerns about the hosts on spamming blacklists.

We concur with these concerns, which we note in Section 5.6, however, we disagree that rate limiting is the simplest explanation. These

forms typically do not allow a user to control the content of the

message, so it does not seem attractive to spammers.

Reviewer A expressed concern about the simplicity of our system. We

would like to reiterate that the key goal of the system is to perform

a wide-scale crawl. To do this, we made explicit performance tradeoffs

to not use Selenium and instead use Soup.

We reconstruct the email forms explicitly to avoid making additional

requests to the same web page, which would incur an additional HTTP

request/response for every fuzzing attempt. We believe this is an

unacceptable overhead, as our goal is to perform a large-scale crawl.

However, Reviewer A brings up an interesting point that this could

bypass CSRF-enabled forms, which would make an interesting extension

to our system.

Reviewer B identifies that the "attack is known for a while and

already tested by many security scanners." We agree, however, security

scanners are closed-source and do not publish their techniques. Just

as SecuBat (WWW 2006) was not the first web vulnerability scanner, it

is incredibly important for the research community to document and

study these techniques. Reviewer B also correctly mentions that "The

attack itself is very straight forward and has known solutions: escape

inputs." Cross-Site Scripting (XSS) and SQL Injection are also

straight forward and have known solutions of escaping input, yet

receive considerable research attention. Our goal is to study the

neglected e-mail header injection vulnerability.

Reviewer B also asks for "a stronger technical contribution like a

code source analyzer ... or looking at hard to exploit injection such

as the one in the body." A static analysis tool for e-mail header

injection is interesting, however, without knowledge of the prevalence

of these vulnerabilities effort on such a tool might be wasted. Our

approach does not require source code. We also note that injection in

the body allows an attacker to control part of the message of the

e-mail but not the recipient, and injecting into headers, which we

study, is more powerful.